An Object Oriented Approach to Content Planning for Text Generation

Ursula Wolz[1]

Columbia University
Department of Computer Science
New York, New York 10027
(212) 854 - 8124
email: wolz@cs.columbia.edu

Abstract

This paper describes GENIE, an object-oriented architecture that generates text with the intent of extending user expertise in interactive environments. Such environments present three interesting goals. First, to provide information within the task at hand. Second to both respond to a user's task related question and simultaneously extend their knowledge. Third, to do this in a manner that is concise, clear and cohesive. Instead of generating text based solely on either discourse goals, intentions, or the domain, we found a need to combine techniques from each. We have developed an object oriented architecture in which the concepts about which we talk (domain entities), the goals that may be accomplished with them (intentions), and the rhetorical acts through which we express them (discourse goals) are represented as objects with localized knowledge and methods. This paper describes how current text planning methods were insufficient for our needs, and presents our object-oriented method as an alternative.

1. Introduction

A practical problem for text generation is how to produce good advice for users of interactive environments. In such settings users attempt to accomplish tasks by combining the set of available commands in a potentially infinite number of ways. In this setting, the text generation problem amounts to describing and justifying a plan of action to the user, or describing trade-offs between alternative plans, for example when suggesting a better way to do a task. In developing GENIE [Wolz *et al.* 90, Wolz 90a, Wolz 90b], an advice giving system for accomplishing tasks using Berkeley Unix Mail, we discovered that current approaches to content planning were insufficient for our needs in two ways. First, to produce the kind of informative responses we found in naturally occurring settings, we needed to carefully partition how textual, domain, and intentional knowledge influence the generation process. Second, in defining textual structure, we found that both schemas [McKeown 85] and simple plan operators [Hovy 88, Moore & Paris 89] could not model the general structures we required without introducing far too many special case operators. Instead we found we could eliminate special case operators by using a two stage process of liberal, minimally constrained planning, followed by pruning in which speech acts *contend* for inclusion in the text.

Both partitioning and our two stage planning process fall naturally within an object oriented paradigm. Objects of our three types, namely domain, intentional and textual, have "knowledge" of how they participate in developing a response to a question. This allows domain knowledge to play more of a role in structuring response content than in other systems. More importantly, as this paper will show, all three can have localized knowledge of how they interact with other objects which reduces the need for special cases.

The next section will elaborate on the practical problem of advice giving in open-ended settings, define our goals for text generation in such settings, and justify why our goals are appropriate. Section 3 shows why schema and plan operators are insufficient. Our object-oriented approach is described in section 4, with an emphasis on text structuring. Section 5 reports on the status of GENIE, and section 6 presents our conclusions.

2. Background

The GENIE project at Columbia addresses the problem of how to extend users' expertise in interactive environments. A communicative problem arises in such environments because users tend to get stuck in a *starter set* [Finin 83] of commands and never progress to more sophisticated methods for accomplishing tasks. Question

[1]Supported in part by ONR grant N00014-82-K-0256, by NSF grant IST-84-51438, a grant from DARPA, and a grant from Siemens Research and Technology Laboratories.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1990		**00-00-1990 to 00-00-1990**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
An Object Oriented Approach to Content Planning for Text Generation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Computer Science Department,Columbia University,New York City,NY,10027**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**10**	

answering in such an environment requires that a text generator not only *respond* to the specific question asked, but also attempt to add a little extra information in order to *enrich* the user's knowledge. The domain we consider is Berkeley Unix Mail, chosen because it is a microcosm of Unix, providing extremely rich and sophisticated functionality with a frustratingly hard to learn interface. The domain also allows us to focus on extending system expertise rather than domain problem solving since the activities, sending, receiving and managing messages, do not require high level thinking skills.

Interactive environments can be characterized as computational tools that allow users to accomplish tasks. Examples range from desk top aides such as word processors, message systems and spread sheets, to cad/cam systems and multi-purpose programming environments. Typically a core set of functionality can be combined and manipulated to construct solutions to domain specific problems. A user approaches such an environment with a domain specific or *computational* goal, and constructs a plan to satisfy that goal with the functional mechanisms available in the environment.

In such open-ended settings no two users will approach the system with the same background or needs. Consequently the degree of difficulty of functionality is more dependent upon the particular experience of a user, than on broad categories of functional difficulty or user expertise [Chin 88]. Users tend to develop their own highly personal repertory of commands. For example, an "expert" user who has extensive experience reading messages, and a "novice", one who has almost no experience reading messages, may both need to be introduced to a method for storing mail in files. The level of expertise of the first user is less significant than whether his or her particular expertise contributes to how much more easily the new method can be communicated. Therefore a particular goal may be satisfied by more than one plan, where the criteria for which plan is "best" is dependent upon the context of the task at hand. This context can be viewed as a model of the user, and it is the primary influence on the *choice* of content and the *structure* in which that content appears as text. It consists of:

- **The discourse context** — The type of question the user is asking, and the specificity of the question. For example, is the user asking for a plan, for an explanation for why a plan failed, or for a better plan than the one he or she is currently using[2].
- **The situational context** — The "physical situation" the user is in, that is, the state of the environment and the user's position in it. This includes background knowledge about domain entities such as what some one's email (electronic mail) address is.
- **Beliefs about the User's Task Expertise** — The computational goals GENIE believes the user knows about and the plans the user has for satisfying them.

Finally, since a user's primary goal is to complete the task, and only secondarily to get new information, he or she would prefer a succinct, informative answer. However the answer should also be sufficiently *grounded*, that is, it should contain *advanced organizers* [Gagne & Briggs 79] that explicitly articulate any of its goals, assumptions, and decisions in choosing a course of domain action that are not obvious to the user.

An example scenario might help illustrate these points. Consider the user model defined at the top of Figure 2-1. The user asks the question, "how do I send a message", which implies a *question intent* to receive a plan for the goal "send a message." The discourse context contains the question intent "given a goal give me a plan", and the expectation that such a plan exists, and that GENIE will provide it to the user. The statement of the goal further indicates that the user is asking about sending a single message, in contrast to "sending mail" where the quantity of messages is ambiguous. The goal statement does not however does not say anything about who the recipient might be. The details of the goal statement can have a profound effect on how GENIE selects content, and chooses to present it. Physically, the user is in the Unix shell, and has no new mail. In the past the user has never initiated the sending of a message, but has only sent messages in reply to messages s/he has received. Finally, as background, in Berekely Unix mail, one cannot "get into" the mail environment with the command "mail" unless one has new mail. The text GENIE produces as a result of this scenario appears in the lower part of the figure.

Sentences 1 - 3 are produced as a result of responding to the user by introducing a plan for sending mail. Sentence 1 is an advanced organizer, which makes sure GENIE is answering the question the user asks. Note that if GENIE had true dialogue capabilities, it might pose this as a question. Only if the user responded in the affirmative would it proceed with the remainder of the text as is. Sentences 4 and 5 are enrichment, elucidating why the user's usual plan won't work. Note too, how discourse goals are merged. For example, sentence 3 is both an example and a definition of syntax.

Text generation for interactive environments therefore requires addressing the following goals:

1. Provide information about the relationship between

[2]A more accurate label might be the "question context," however eventually this structure should represent the dialogue between the user and GENIE over time. We keep the more grand label to remind ourselves of this.

Input:
Discourse: How can I send a message to message?
Situation: User is in Unix, with no new mail.
Kathy is a user with email address "kathy."
User knowledge of domain tasks:
User has never sent a message from Unix shell, but has only initiated sending from "read mode" after reading new mail.
Output:

(1) I assume you are in unix, your recipient is an individual, and the address is a local address. *(2)* You must supply the email address of the individual. *(3)*For example, to send mail to Kathy, type

mail kathy

since kathy is her email address.
(4) You usual method is to read your mail, then use the mail command inside the mail environment. *(5)* Since you do not have any new mail, your usual method will not work.

Figure 2-1: An example scenario

users' task (their computational goal) and the methods (plans) that may be used to accomplish them. Do this within the current context of the task at hand.

2. Respond informatively, but also attempt to provide enriching material.

3. Provide advanced organizers, but do it in a manner that is concise, clear and cohesive.

Taken together these goals suggest that a large amount of information must be presented in as short a text as possible. Evidence from three informal studies indicates that natural occurring texts satisfy these goal. First a set of textual materials, including manuals, tutorials, texts and canned text on-line resources were studied. Tutorials and on-line resources tended to focus on *responding* directly to the problem of how to do tasks. Reference manuals, text books and especially "advanced user" manuals focused on *enrichment*. In all, 20 passages covering information about Unix, Lisp, Pascal, Logo and a number of word processors were analyzed. These texts have strongly influenced our approach to content selection and structuring. In particular, we found that four types of strategies occurred through which information about the relationship between domain plans and goals was given:

- **Introducing:** Presenting plans that the user has not encountered before.
- **Reminding:** Briefly describing plans to which the user has been exposed, but may have forgotten.
- **Clarifying Distinctions:** Explaining distinctions and options about plans.
- **Elucidating Misconceptions:** Clearing up misunderstandings that have developed about plans to which the user has been exposed.

These strategies had a consistent general structure that included sub-strategies such as summarizing or elaborating on a plan that had their own linguistic structure. See [Wolz *et al.* 90] for a thorough discussion of these strategies. The "Introduce" strategy will be discussed extensively in the following sections.

The second analysis looked at 30 unix related question answering sessions in which the correspondence occurred through electronic mail. We found that the responses still fell within the four strategies. The respondent made an attempt to respond informatively, and in 8 cases explicitly included contextual information that acted as an advanced organizer, or that might not have been obvious to the questioner. In 7 cases the respondent included information that was not merely in response to the question, but that could extend the questioner's expertise. Most importantly however, we found instances where sub-strategies, and sometimes the four basic strategies themselves, were combined to produce very concise utterances. We also noticed that sub-strategies were absent when their contents was not critical, or when it was obvious from the question. The last of these points have led us to the goal of conciseness.

A third study of human-to-human tutoring in a computer lab also corroborated the strategies, the incidence of enriching behavior, the use of advanced organizers. However, verbal responses tended to be significantly more long winded, exhibiting more of a stream of consciousness than the mail messages. Since GENIE produces interactive text, our goal is to most closely match the mail message behavior, and aim for conciseness.

3. Limitation of Current Approaches

GENIE was developed as a result of the design goals enumerated above, namely to answer a question within the task at hand, satisfy the dual discourse goals of responding and enriching, and do this clearly and concisely. These goals presented some special problems for capturing textual structure, and for defining choice points for refining that structure into natural language clauses and vocabulary.

In order to cover two simultaneous discourse goals through clear and concise text, we discovered that schemas [McKeown 85, Paris 87] provided too rigid a structure with insufficient definition of choice points, while plan operators [Hovy 88, Moore & Paris 89] provided too many choice points without sufficient structure. In order to answer a question within the task at

hand, GENIE needed to make choice point decisions based on *intentional* [Appelt 85, Moore & Paris 89], *domain* and *textual* [McKeown 85, Paris 87] knowledge. Previous work on text generation has tended to put the emphasis on one of the three, incorporating the other two implicitly.

The problem can be best illustrated by looking at the informal description of the strategy of "introducing" presented in Figure 3-1. The structure of this strategy was derived from principles of instructional design [Gagne & Briggs 79] and on the analysis of text. The informal definition contains both structure, in the form of how to expand an introduction into sub-structures such as "summarize", and under what circumstances to do so, for example whether the goal is satisfied by an action or another plan, or whether the user knows or does not know the subgoal. This strategy assumes a recursive definition of a goal, where the goal can be satisfied by an action or indirectly by a plan that itself consists of a set of subgoals. This definition of a goal follows that used in STRIPS [Fikes and Nilsson 71] where the term goal refers to a *state difference*. It is to be distinguished from the *goal state*, or *end state* which is sometimes referred to as "the goal."

Casting this informal description as a formal schema looses the explicit information about what circumstances motivate expansion into particular sub-structures. For example, the schema definition presented in Figure 3-2 does not contain choice point knowledge.

Conversely, casting the description as a set of plan operators looses the explicit structure in the informal description. The structure emerges only during planning. Researchers in text generation using plan operators (Appelt85,Hovy88, Moore-paris89) have adapted the hierarchical definition of a plan operator developed by Sacerdoti [Sacerdoti 77] A discussion of the merits and disadvantages of the subtle differences between definitions is not appropriate here. Instead, we must focus here on the components of plan operators that are critical to representing choice points and sub-structure, namely a constraint/action-body pair. *Constraints* or preconditions define choice points, while the *body* of the operator defines actions that are taken if the constraints are met. Figure 3-3 shows informal definitions of some of the operators that would be required to model the sub-structures that appear in Figure 3-1.

Plan operators introduce an even more insidious problem. Note that both Introduce-plan and Introduce-action contain the operator "state-goal" in their bodies. Other high-level strategies such as Remind Plan or Elucidate Misconception also include this sub-structure. Recall that GENIE will initiate two strategies, one in response and one to enrich. For example, to produce the text in Figure 2-1 GENIE chose to respond by *introducing* a new plan to the user, and enrich, by *elucidating* why the user's usual plan won't work in the current situation. The fundamental

1. Informally, introducing a plan consists of
   a. Stating the goal.
   b. If the goal is satisfied by an action, introducing the action, otherwise:
   c. Stating any assumptions that affected the choice of plan.
   d. Summarizing the sub-goals for the plan.
   e. For each sub-goal either introducing or reminding about the plan for the sub-goal depending on whether the model of user task expertise indicates the user knows how to satisfy the sub-goal.
   f. If the plan is not the top-level plan, reviewing the steps in the plan through an example.
   g. Relating each step in the example to a sub-goal.
2. Introducing an action consists of:
   a. If the goal which the plan satisfies is the top level goal, stating the goal.
   b. Presenting the syntax.
   c. Describing the parameters.
   d. Describing any preconditions that must exist for it to work.
   e. Describing the effects (which is not the same as stating the goal).
   f. If the goal which the plan satisfies is the top level goal, giving an example.

Figure 3-1: Introducing a plan or an action

Introduce Schema
{goal-statement}
{assumptions plan-summary
(introduce | remind) example* comparison*}
{syntax parameters preconditions
effects example*}

{} = optionality
* = zero or more

Figure 3-2: The strategy of introducing as a Schema

problem with having two discourse strategies within a single text is that when each is instantiated as a sequence of speech acts, redundancy and conflicts may appear unless the instantiations of those structures are collaboratively developed. Furthermore, in GENIE, the highest level strategies (remind, introduce, elucidate, clarify) all contain at least one instance of recursion on another top level strategy. For example, introducing a plan requires introducing or reminding the substeps of the plan. Published descriptions of current content planners

introduce-plan
constraints: goal is satisfied by plan
action-body: state-goal
state-assumptions
summarize-plan
expand-plan
plan-example
relate-steps

introduce-action
constraints: goal is satisfied by action
action-body: state-goal
state-syntax
state-preconditions
state-parameters
action-example

Figure 3-3: Possible plan operators for introducing

don't show how this kind of open-ended recursion is handled. Examples tend to only show how a top level discourse goal is elaborated through a refinement process. When runaway recursion might occur, computational tricks such as limiting the number of levels of deepening is suggested. This is not sufficient for GENIE's purposes.

For example consider a domain goal G that is satisfied by plan P, that expands into subgoals, that are satisfied by plans that themselves expand until a set of actions is reached. Clearly one can't limit the expansion since the actions are a critical component of the text. If one blindly follows the informal description of introducing the top level plan P, then all of the subgoals and sub-plans will be included recursively. Merely stating the top level goal and plan, and the resulting actions would seem to be a natural intuitive solution. But this would require different operators for "high" level strategies, than for those further down the structure. This thwarts the elegant nature of independent, generalized structures such as schema's and plan operators. To further aggravate the situation, consider that many of those actions may have the same preconditions. Again, if one blindly follows a generalized "introduce action" operator, then for all those actions that include precondition P, a speech act stating P will be included. One would prefer that P be stated at most once.

Avoiding this problem through "special case" operators is insufficient. For example, consider two plan operators that handle when to state the goal.

1. If introducing in response, include a statement of the goal unless it is explicitly mentioned in the question.
2. If introducing as enrichment, do the same thing except if you already mentioned the goal in response, don't mention it again.

The second operator is dependent upon the result of the first one, and although superficially they seem to be independent entities, they are not. Now a third special case is introduced when "elucidating a misconception", because here too, there are circumstances in which the goal should be stated. All three operators know about each other implicitly through their constraints. Because plan operators based on Sacerdoti's hierarchical model assume a set of abstraction spaces, "high level" operators always precede lower-level ones, and the higher ones consequently have an implicit influence on the subordinate ones.

A final problem with implementing the informal structure of Figure 3-1 as plan operators is that a significant amount of domain specific knowledge is embedded in the operators. The operators in Figure 3-3 illustrate this point. The "state-..." operators are essentially "inform" speech acts, but they imply that a specific sort of domain object is to be the subject of the informing. Although text plan operators are supposed to be textual in nature, it is far too easy to implicitly embed domain knowledge in them. This compounds the complexity in developing constraints because textual, domain, and intentional knowledge can be haphazardly intertwined. A similar problem occurs in creating the tests on schemas, since the test may be described by an arbitrary lisp expression. In developing GENIE, we found ourselves mired in layers of operators that contained such complex inter-related constraints. This suggested a different sort of architecture that would allow us to deal with the complexity of inter-related, but distinct influences, and at the same time model both structure and choice of structure in one formalism.

4. An Object-Oriented Architecture for Content Planning

Given the problems described in the previous section, we will focus here on the aspects of GENIE's object-oriented architecture that address the problems of text structuring and content choice. The over-all architecture is described in [Wolz 90a]. In particular, this section will illustrate how special case operators are avoided by cleanly dividing the structuring process into two stages. First, rhetorical strategies are refined with minimal constraints to the point of rhetorical acts. Second, the rhetorical acts *contend* for inclusion in the set that is realized as text. Some acts may have cause to be explicitly included or excluded, and among those that are included some may be merged with others, while some may need to be explicitly kept separate. Before these processes can be described however, it is necessary to articulate the nature of object-orientedness as it applies to GENIE, and describe the input to the text structuring phase.

4.1. Object Classes for Content Planning

GENIE employs three overlapping classes of objects; domain objects, intentional objects and discourse objects. *Domain Objects* classify domain entities, including both the commands such as "mail" or "copy" that the user invokes, and the entities manipulated by commands such as files, messages and users. *Intentional Objects* allow GENIE to construct and analyze plans, and include computational goals, plans to satisfy them, and actions (which contain commands as a subclass). *Discourse Objects* include rhetorical strategies, rhetorical plans, rhetorical acts, syntactic structures and vocabulary. Note that these classes are not mutually exclusive. For example, a goal such as "wanting to send mail" is both an intentional and domain object since it is a domain specific goal.

Each object class has explicit relationships to other classes. For example the class "goal" contains an attribute "who-satisfies-me", whose value can point to either a plan or an action. A plan in turn is an object class that is related to a rhetorical strategy "talk-about." The strategy in turn knows under what circumstances the object about which it talks (the plan) should be introduced, reminded about, clarified or elucidated, that is, it is related to these other rhetorical strategies. Finally, rhetorical strategies may be instantiated either by *rhetorical plans*, that recurse to other strategies, or by *rhetorical acts* such as "stating" that can be manifested as English text.

4.2. Constructing Domain Objects for Inclusion in the Text

Recall that GENIE answers questions about how to do tasks. In other words, it provides descriptions of the relationships between goals, plans, actions and the effects of actions in the domain. Providing an answer involves understanding the user's question in a situational context by instantiating a set of domain objects, generating the set of objects (domain goals, plans, actions, effects) that are expected in reply, and selecting the discourse objects that talk about those objects.

For example, consider the question from Figure 2-1: "How do I send a message?" As a result of parsing this sentence, the discourse context will include an instantiation of the very specific goal "send-mail." The discourse context will also include the *question intent* that the user expects to be told the *best plan* for the goal in the current context. A specialized plan class knows how to construct an instantiation of such a plan, calling it `best-plan`. It creates a relationship between the instantiation of the "send-mail" goal, and `best-plan` and relates `best-plan` to an instantiation of an abstract plan for satisfying that goal. It also instantiates *discriminator objects* that capture the assumptions and decisions that were made in constructing the abstract plan. Like other objects that might be included in the final text, these discriminator objects, which are domain dependent, have rhetorical strategies that can talk about them. `best-plan` is then refined, by instantiating the subgoals of its abstract plan until the process bottoms out as actions.

Figure 4-1 shows some of the itentional objects that are instantiated for `best-plan`. Describing the actual process here would take us too far afield. Note that the discriminators include which part of the user model affected the choice. Decisions that are based on the discourse and situational context are "stronger" than those that are based on the model of user task knowledge. The former are deduced from *facts*, the latter, like default heuristics for object types, are deduced from weaker *beliefs*. The strength of the deduction will affect how the decision is described in the text.

BEST-PLAN (plan)
subgoals: ENTER-SEND-MODE-1, CHOOSE-RECIPIENTS-1

ENTER-SEND-MODE-1 (goal)
discriminator: UNIX-READ-1
who-satisfies-me: B-SUB-1

B-SUB-1 (plan)
subgoals: SEND-FROM-UNIX-1

SEND-FROM-UNIX-1 (goal)
who-satisfies-me: SEND-FROM-UNIX-ACTION-1

CHOOSE-RECIPIENTS-1 (goal)
discriminator: GROUP-SINGLE-1
who-satisfies-me: B-SUB-2

B-SUB-2 (plan)
subgoals: CHOOSE-SINGLE-1

CHOOSE-SINGLE-1 (goal)
discriminator: LOCAL-REMOTE-1
who-satisfies-me: B-SUB-3

B-SUB-3 (plan)
subgoals: CHOOSE-LOCAL-1

CHOOSE-LOCAL-1
who-satisfies-me: CONSTRUCT-LOCAL-ADDRESS-ACTION-1

UNIX-READ-1 (discriminator)
choice-from: SITUATION

GROUP-SINGLE-1 (discriminator)
choice-from: USER-TASK-MODEL

LOCAL-REMOTE-1 (discriminator)
choice-from: USER-TASK-MODEL

Figure 4-1: Set of Objects to be Included in Response

The best plan by itself is not sufficient for GENIE's design goals. In particular, its construction offers little insight into what the user knows about it, or for that matter whether the user knows any plan for satisfying the goal, and whether that plan works in the current context. Therefore, in this context it is also necessary to construct a *User Model Plan*, `user-plan`, that has a similar structure. The two plans are "compared" by annotating differences in the sets of objects produced for each. More

importantly, the best-plan and user-plan are annotated as to whether each is valid in the current situation, and if so, whether they are the same plan. The information is critical to how each will be "talked about." In the scenario of Figure 2-1 the user-plan will not work because it requires that new mail exist, which is not the case in the scenario. Once the relationship between the goal, best-plan and user-plan have been established, text structuring can begin.

4.3. Structuring Content: Instantiating Rhetorical Strategies

Structuring content consists of selecting two rhetorical strategies, one for responding, and one for enriching, and instantiating a rhetorical plan for each strategy. The selection is based on the nature of the question intent, the domain plans that have been constructed, whether those domain plans satisfy the goal, and whether they are identical. The selection rules were compiled from our design goals, and result in one plan being described in response and possibly a second being described as enrichment. The plans for the two strategies are then recursively expanded until the process bottoms out as a set of rhetorical acts.

Structuring begins when the domain plans such as best-plan and user-plan are sent messages to "talk-about" themselves[3]. In the example scenario, best-plan decides that it should be "introduced in response to the question" because it is not identical to user-plan. Had the plans been identical, best-plan would have chosen to be reminded instead. user-plan decides that it should be "elucidated as enrichment" because it doesn't work in the current context.

The strategies map to rhetorical plans that are a sequences of message passing instructions with simple control structures, such as interating on a list, and binary branching. A rhetorical plan is refined by sending those messages to other rhetorical strategies or to rhetorical acts that cannot not be further refined. The method for expanding the rhetorical plan is dependent upon the object type of the entity to be talked about. Classes of objects may use the same rhetorical strategy through inheritance, however, when appropriate two classes may require the same strategy, but have their own private rhetorical plans for those strategies. For example, given the informal descriptions of introducing a plan or an action in Figure 3-1, it is clear that the text structure of introducing these two kinds of things is dissimilar. The rhetorical strategy associated with introducing a plan is formally described as:

```
(rh-plan introduce plan
  ((state-it goal-I-satisfy)
   (ITERATE choice ON choices-made WITH
     (describe-decisions choice)
   (ITERATE subgoal ON subgoals WITH
     (summarize subgoal))
   (ITERATE subgoal ON subgoals WITH
     (IF (subgoal in-user-model)
       (remind who-satisfies-me)
       (introduce who-satisfies-me)))))
```

This plan says, to introduce a plan, state the goal it satisfies. For each choice made to construct it, describe the reason for the choice. For each subgoal, summarize the subgoal, then for each subgoal, if the subgoal is known by the user, remind the user about how it can be satisfied, otherwise, introduce how it can be satisfied. A particular object, like a domain plan, has a set of attributes or *slots* that link it to other objects. For example, a plan has a "goal-I-satisfy" slot that relates it to the goal it satisfies, and the goal in turn has a "who-satisfies-me" slot that points to the plan. The statement "(state-it goal-I-satisfy)" says to send a "state-it" message to the object related to me by my "goal-I-satisfy" attribute. When this message is sent, a rhetorical object of class "state-it" is instantiated that links the calling rhetorical object with the called object, creating ties between them and the domain objects.

Of particular significance is the fact that rhetorical plan are minimally constrained. For example, the plan for introducing a domain plan has no constraints on when to include the statement of the domain goal, so this plan can be used both to describe the best-plan, or one of its sub-plans, or the user-plan without complex constraints. More importantly, this rhetorical plan does not need to keep track of which operators are appropriate for different kinds of objects. For example, when expanding its subgoals, it merely sends a message to the object related to the subgoal by the "who-satisfies-me" slot. The object, either a plan or an action, upon receiving the message instantiates its localized rhetorical plan for reminding or introducing. This process continues until a set of rhetorical acts is constructed.

4.4. Contending for Inclusion: Communication Between Rhetorical Acts

The structuring stage produces an initial ordering of rhetorical acts that contain functional descriptions which when given to the surface generator can produce text. The form of these descriptions is based on the theory of functional grammar [Halliday 85]. Due to the minimally constrained refinement employed during the expansion of the strategies, the set of rhetorical acts produce redundant, extraneous or obvious information that would be terribly verbose if instantiated as text. For example, Figure 4-2 shows the text that would be produced if the first 6 rhetorical acts instantiated in the send mail example were

[3]In other scenarios, other kinds of plans might be constructed, see [Wolz 90a] for details.

given to the surface generator. Note that these 6 acts are only 1/5 of the rhetorical acts that are produced in this example.

```
STATE-IT-GOAL-1 ::
 domain-object[goal]:send-mail-1
STATE-IT-STRONG-DISCRIMINATOR-1 ::
 domain-object[discriminator]:unix-read-1
STATE-IT-GOAL-2 ::
 domain-object[goal]:choose-recipients-1
STATE-IT-WEAK--DISCRIMINATOR-1 ::
 domain-object[discriminator]:group-single-1
STATE-IT-GOAL-3 ::
 domain-object[goal]:send-individual-1
STATE-IT-WEAK-DISCRIMINATOR-2 ::
 domain-object[discriminator]:local-remote-1

You want to send mail.
I know you are in Unix.
You want to specify a recipient.
I assume your recipient is an individual.
You want to specify a single recipient.
I assume the address is a local address.
```

Figure 4-2: Text Produced Before Content Filtering

The contention stage reviews this list and specifically includes, excludes, merges or keeps separate acts within this list. For example information derived from either the situational context, such as the mode the user is in, should be included because it may not be obvious to the user. Information derived from the discourse context, such as an explicit reference to the goal may be excluded as obvious because it was referred to in the question. In the process of both reminding and clarifying, the same preconditions may be introduced twice. These can be merged into one utterance. However, if a component of an utterance is critical, such as a failed precondition when elucidating a misconception, then it should be kept separate.

The rhetorical acts contain the methods for determining how they can be filtered. This includes specific rules for when they may be excluded, when they must be included, what other objects they may or must be merged with and which one is dominant, and when they are to be kept separate. Contention for inclusion in the text begins when the set of rhetorical acts is sent messages to "execute" themselves. Each examines its methods for how it may be merged etc. Processing continues until all objects return successfully. Some objects will have been excluded and some will have instantiated new "merged" objects, subordinating themselves to the merged object. Restructuring occurs during merges and excludes. A merge may move an act up or down depending on whether the dominant act comes before or after the submissive one. Two acts which are initially separated by some distance may find themselves next to each other, or even merged, if the acts between them are excluded. Once processing is complete, the remaining objects apply knowledge for constructing clause level structures which are then passed to the surface generator.

Figure 4-3 shows how the 6 acts from Figure 4-2 contend to be included, and are eventually merged to produce the first sentence in Figure 2-1. The first act of stating the main goal can be excluded because it is in the discourse, and is probably obvious to the user. The second and third statements of goals may be excluded because they are part of a goal-plan-goal chain, that is, they supply unnecessary detail about the relationship between the top level goal and the actions. The result of these deletions makes the two "weak" discriminator statements adjacent, at which point they can be merged. This newly merged object can then be merged with the "strong" discriminator that follows it. In general, when weak assertions out number strong ones, the weak assertions can subsume the strong assertions, if the significance of the strong assertion isn't important. For example, it is acceptable to say "I assume X and Y" even if you "know X" provided the distinction between knowing X and assuming X is not critical to the response.

```
Excluded!
 STATE-IT-GOAL-1 because IN-DISCOURSE-CONTEXT
Excluded!
 STATE-IT-GOAL-2 because IN-GOAL-PLAN-GOAL-PATH
Excluded!
 STATE-IT-GOAL-3 because IN-GOAL-PLAN-GOAL-PATH
Merged!
 STATE-IT-WEAK-DISCRIMINATOR-2 and
 STATE-IT-WEAK-DISCRIMINATOR-1 into
 CONJOINED-WEAK-DISCRIMINATOR-1 because
 SAME-TYPE
Merged!
 CONJOINED-WEAK-DISCRIMINATOR-1 and
 STATE-IT-STRONG-DISCRIMINATOR-1 into
 CONJOINED-WEAK-DISCRIMINATOR-2 because
 WEAK dominates STRONG

I assume you are in unix, your recipient is an individual, and
the address is a local address.
```

Figure 4-3: Application of Filtering Methods

5. Status

GENIE is implemented on a Sun 3/60 in Sun Common Lisp. All of the object types described are represented using Hyperclass [Smith and Carando 86] which provides powerful inheritance mechanisms. Surface text generation is accomplished through FUF [Elhadad 88] that employs a two stage process of functional unification [Kay 79] and linearization to produce English

text. All of the functionality of Berkeley Unix Mail can be represented with the exception of command files which allow customization by loading directives into the mail environment.

Our approach provides a large degree of flexibility. For any single computational goal, at least 120 scenarios can be generated based on the variability between the discourse context, the situational context and the model of user task expertise [Wolz 90b]. It remains to be seen whether this range significantly impacts the range of text that is generated. We are about to conduct an analysis of the relationship between the text produced and the input scenarios. We expect to verify statistically that the occurrence of specific rhetorical acts in the text can be predicted by the information included in the input. Furthermore we expect to show that across texts, the number of sentences produced is considerably smaller than the number of unmerged rhetorical acts.

6. Conclusions

Three open questions remain. First, can the user model required as input to GENIE be produced automatically? Second could GENIE be extended to enable it to engage in an extended dialogue? Third, how combinatorially expensive is the contention process?

Building and maintaining the user model would involve developing a system that can observe user behaviors and draw inferences about user knowledge from those behaviors. There are two obvious observable behaviors, the questions asked by the user, and the actions taken that affect the situation. Part of the solution to maintaining the user model may be found in extending the communicative ability of GENIE to allow it to engage in an extended dialogue. In order to do so, GENIE would need to be able to maintain a history of the discourse context, the situational context in which the preceding discourse took place, and the reasoning it used to choose a response. With such data, comparisons could be made about what entities were talked about or used over time, allowing conclusions about user knowledge to be drawn. Currently GENIE contains the requisite knowledge representations, we are confident that the appropriate knowledge acquisition systems can be built.

Another question is the problem of mapping GENIE's rhetorical acts to surface forms. In the present implementation the functional descriptions embedded in the acts is extremely primitive. Although FUF supports lexical choice, it is not presently exploited in GENIE, and although we make use of complex sentences, there is no principled theory for our choice of sentence structure. Work by Elhadad [Elhadad 90] is directed toward these problems.

Finally, it has yet to be determined how expensive the contention phase of structuring is. The problem is whether the acts must be examined more than once, and if so how often. We suspect that since the acts are represented as objects this won't prove to be intractable. general. This is because examining act 1 in relation to act 2, "marks" both acts, so 2 doesn't have to examine 1 in return. This sort of marking is particularly important in merging two objects. Furthermore the methods restrict the classes of objects that may be examined when trying to merge (for example, a syntax-statement doesn't try to merge with a goal), so the number of comparisons is further reduced.

To summarize, in this paper we have shown how knowledge of the domain, intentions and discourse can be combined in an object oriented architecture for text planning. In particular we have focused on the problem of how to support recursive interaction between plan operators, because of the complexity of special cases and constraints this requires. Our solution, to use a two stage process of structuring and act contention shows how simpler models can be implemented, and highlights the power of modularization through an object-oriented approach.

Acknowledgments

Kathy McKeown and Gail Kaiser deserve thanks for supporting this work. Dr. McKeown also provided valuable suggestions on the final version of this paper, as did the NL Generation Workshop reviewers. I am grateful to David Robinowitz and Wonsuk Jang for proving that GENIE could be integrated with FUF, and to Michael Tanenblatt for the readability of his code.

References

[Appelt 85] Appelt, D. E. *Planning Natural Language Utterances.* Cambridge University Press, Cambridge, England, 1985.

[Chin 88] Chin, D. N. *Intelligent Agents as a Basis for Natural Language Interfaces.* Ph.D. Thesis, University of California, Berkeley, 1988.

[Elhadad 88] Elhadad, M. The FUF Functional Unifier: User's manual. Technical Report CUCS-408-88, Columbia University, June, 1988.

[Elhadad 90] Elhadad, M. Constraint-Based Text Generation: Using Local Constaints and Argumentation to generate a turn in conversation. Technical Report CUCS-003-90, Columbia University, New York, NY, 1990.

[Fikes and Nilsson 71] Fikes, R. E. and Nilsson, H. J. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial Intelligence* 2, 1971, pp. pages 189-205.

[Finin 83] Finin, T. Providing help and advice in task oriented system. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983, pp. 176-178.

[Gagne & Briggs 79] Gagne, R. M. and L. J. Briggs *Principles of Instructional Design.* Holt Rinehart and Winston, New York, 1979.

[Halliday 85] Halliday, M.A.K. *An Introduction to Functional Grammar.* Arnold, London, 1985.

[Hovy 88] Hovy E.H. Two types of planning in language generation. Proceedings of 26th Meeting of ACL, Association for Computational Linguistics, 1988.

[Kay 79] Kay, M. Functional Grammar. Proceedings of the 5th meeting of the Berkeley Linguistics Society, 1979.

[McKeown 85] McKeown, K.R. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Studies in Natural Language Processing. Cambridge University Press, Cambridge, England, 1985.

[Moore & Paris 89] Moore, J.D. and C.L. Paris. Planning text for advisory dialogues. Proceedings of 27th Meeting of ACL, Association for Computational Linguistics, 1989, pp. 203 - 211.

[Paris 87] Paris, C. L. *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise.* Ph.D. Thesis, Columbia University, 1987.

[Sacerdoti 77] Sacerdoti, E. *A Structure for Plans and Behavior.* American Elsevier North-Holland, New York, 1977.

[Smith and Carando 86] Smith, R.G and P.J. Carando. Structured Object Programming In Strobe. Schlumberger-Doll Research, Ridgefield, CT, 1986.

[Wolz 90a] Wolz, U. *Extending User Expertise in Interactive Environments.* Ph.D. Thesis, Columbia University, 1990. Forthcoming.

[Wolz 90b] Wolz, U. The impact of user modeling on text generation in task-centered settings. Second International Conference on User Modeling, Honolulu, Hawaii, 1990.

[Wolz *et al.* 90] Wolz, U., K.R. McKeown and G. E. Kaiser. "Automated tutoring in interactive environments: A task centered approach." *Machine-Mediated Learning 3*, 1, 1990, pp. 53-79.